

Lab 13 –Threads

Neste laboratório você fará o uso básico de **thread** em Java.

Exercícios

Exercício 1: Herdando a classe **Thread**

Exercício 2: Implementando **Interface Runnable**

Exercício 3: Sincronizando **threads**

Exercício 1 - Herdando a classe Thread

1.1. O método start() sem o construtor da subclasse

1. Crie um projeto Java, depois crie a classe **ImprimeNomeThread.java** conforme código abaixo.

```
// Subclass extends Thread class
public class ImprimeNomeThread extends Thread {

    public ImprimeNomeThread( String nome ) {

        super(nome);

    }

    // Sobrescreve método run() da classe Thread.
    // Este método toma a execução método start() for invocado
    public void run() {

        System.out.println("metodo run() da thread " + this.getName() + " e' chamado");

        for (int i = 0; i < 10; i++) {

            try {

                sleep(1000);
                System.out.println(i + " : " + this.getName());

            } catch (InterruptedException e) {
                e.printStackTrace();
            }

        }

    }

}
```

2. Agora vamos adicionar o método main a classe **ImprimeNomeThread.java**, para que crie execute nossa **thread** e a inicialize chamando o método **start()** conforme abaixo.

```
// Cria instancia de uma classe que e
// subclasse da classe Thread
public static void main(String[] args) {

    System.out.println("Criando instancia de ImprimeNomeThread...");

    ImprimeNomeThread minhaThread = new ImprimeNomeThread("A");

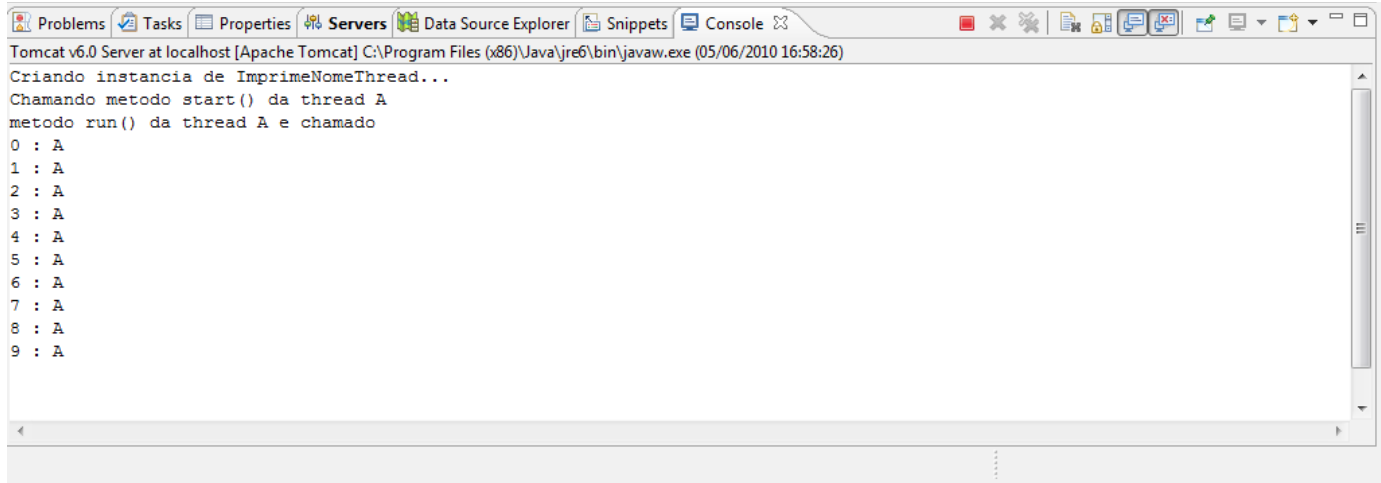
    // Inicia thread pela invocação do método start()

    System.out.println("Chamando metodo start() da thread " + minhaThread.getName());

    minhaThread.start();

}
```

3. Execute o arquivo como aplicação java e fique de olho na **console do Eclipse** o resultado que será impresso. Perceba que enquanto eu não encerro o processo ou termine a execução forçada da aplicação, o sistema não irá para de imprimir os valores na console.



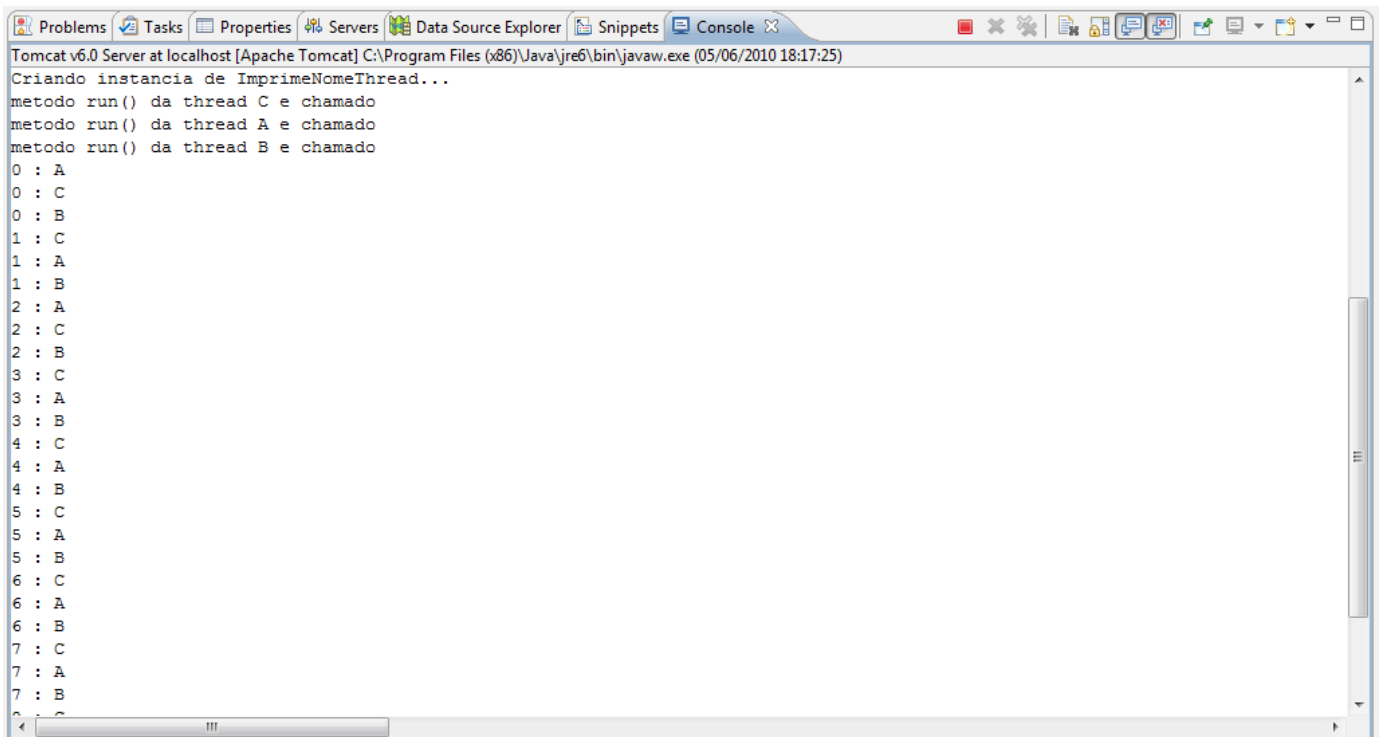
1.2 O método start() está no construtor da subclasse

1. Modifique a thread **ImprimeNomeThread.java**, adicionado ao construtor a chamada ao método **start()** conforme abaixo. Esta alteração dispensa a necessidade de após ser criado o objeto pelo **NEW** tenha que chamar o método **start()** para iniciar, ou seja, a **thread** será criada e inicializada por uma chamada ao seu **construtor**.

```
public ImprimeNomeThread(String nome) {  
    super(nome);  
    //metodo start() dentro do construtor da subclasse  
    start();  
}
```

2. Modifique o método main alterando o código que foi adicionado no exercício 2 anterior para que fique conforme abaixo e execute novamente a aplicação para ver o que irá acontecer.

```
// Cria instancia de uma classe que e  
//subclasse da classe Thread  
System.out.println("Criando instancia de ImprimeNomeThread...");  
ImprimeNomeThread pnt1 = new ImprimeNomeThread("A");  
ImprimeNomeThread pnt2 = new ImprimeNomeThread("B");  
ImprimeNomeThread pnt3 = new ImprimeNomeThread("C");
```



```

Tomcat v6.0 Server at localhost [Apache Tomcat] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (05/06/2010 18:17:25)
Criando instancia de ImprimeNomeThread...
metodo run() da thread C e chamado
metodo run() da thread A e chamado
metodo run() da thread B e chamado
0 : A
0 : C
0 : B
1 : C
1 : A
1 : B
2 : A
2 : C
2 : B
3 : C
3 : A
3 : B
4 : C
4 : A
4 : B
5 : C
5 : A
5 : B
6 : C
6 : A
6 : B
7 : C
7 : A
7 : B

```

Exercício 2 - Implementando Interface Runnable

2.1 - Iniciando fora do construtor da classe que implementa Runnable

1. Agora vamos criar uma thread implementado a interface Runnable, crie a classe ImprimeNomeRunnable.java conforme abaixo.

```

public class ImprimeNomeRunnable implements Runnable {

    String nome;

    public ImprimeNomeRunnable( String nome ) {

        this.nome = nome;
    }

    // implementação do método run() da interface Runnable
    public void run() {

        for (int i = 0; i < 10; i++) {
            // sleep(1000); ERRO de compilação porque não herda Thread
            System.out.println(i + " : " + nome);
        }
    }
}

```

2. Agora vamos adicionar o método main a classe **ImprimeNomeRunnable.java**, para que seja criado threads da classe que implementa Runnable.

```

public static void main(String[] args) {

    ImprimeNomeRunnable pntr1 = new ImprimeNomeRunnable("RA");
    Thread t1 = new Thread(pntr1);
}

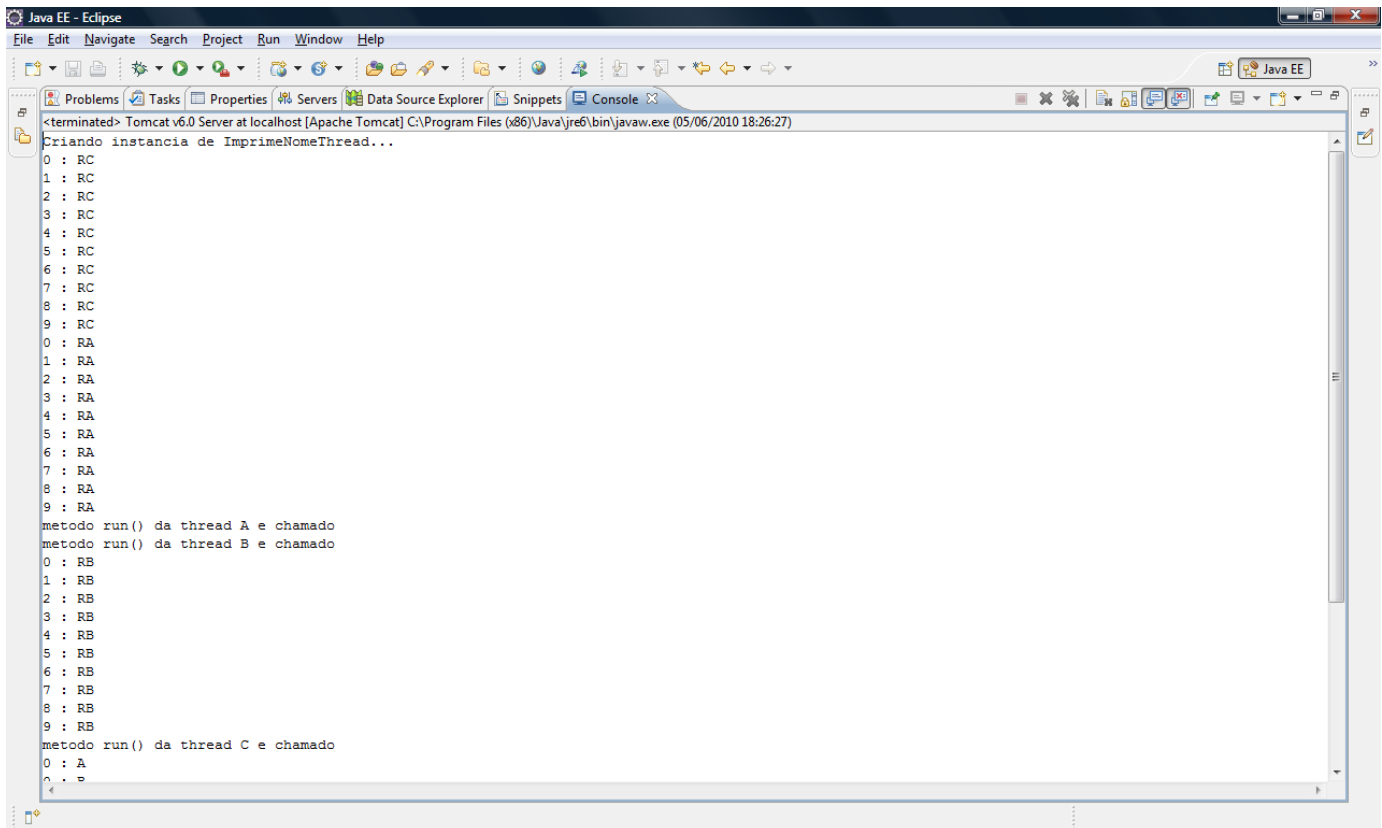
```

```
t1.start();

ImprimeNomeRunnable pntr2 = new ImprimeNomeRunnable("RB");
Thread t2 = new Thread(pntr2);
t2.start();

ImprimeNomeRunnable pntr3 = new ImprimeNomeRunnable("RC");
Thread t3 = new Thread(pntr3);
t3.start();

}
```



2.2 - Iniciando dentro do construtor da classe que implementa Runnable

1. Vamos alterar a classe **ImprimeNomeRunnable.java** para iniciar a thread no momento em que for criado o objeto, através do construtor, da mesma forma que fizemos anteriormente com a classe que herdava de **Thread**. Altere a classe para que fique conforme abaixo.

```
public class ImprimeNomeRunnable implements Runnable {

    Thread thread;

    public ImprimeNomeRunnable( String nome ) {

        thread = new Thread(this, nome);
        thread.start();

    }

    // implementação do método run() da interface Runnable
    public void run() {
```

```

        for (int i = 0; i < 10; i++) {
            try {
                thread.sleep(1000); // Agora é permitido porque é uma thread
                System.out.println(i + " : " + thread.getName());
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

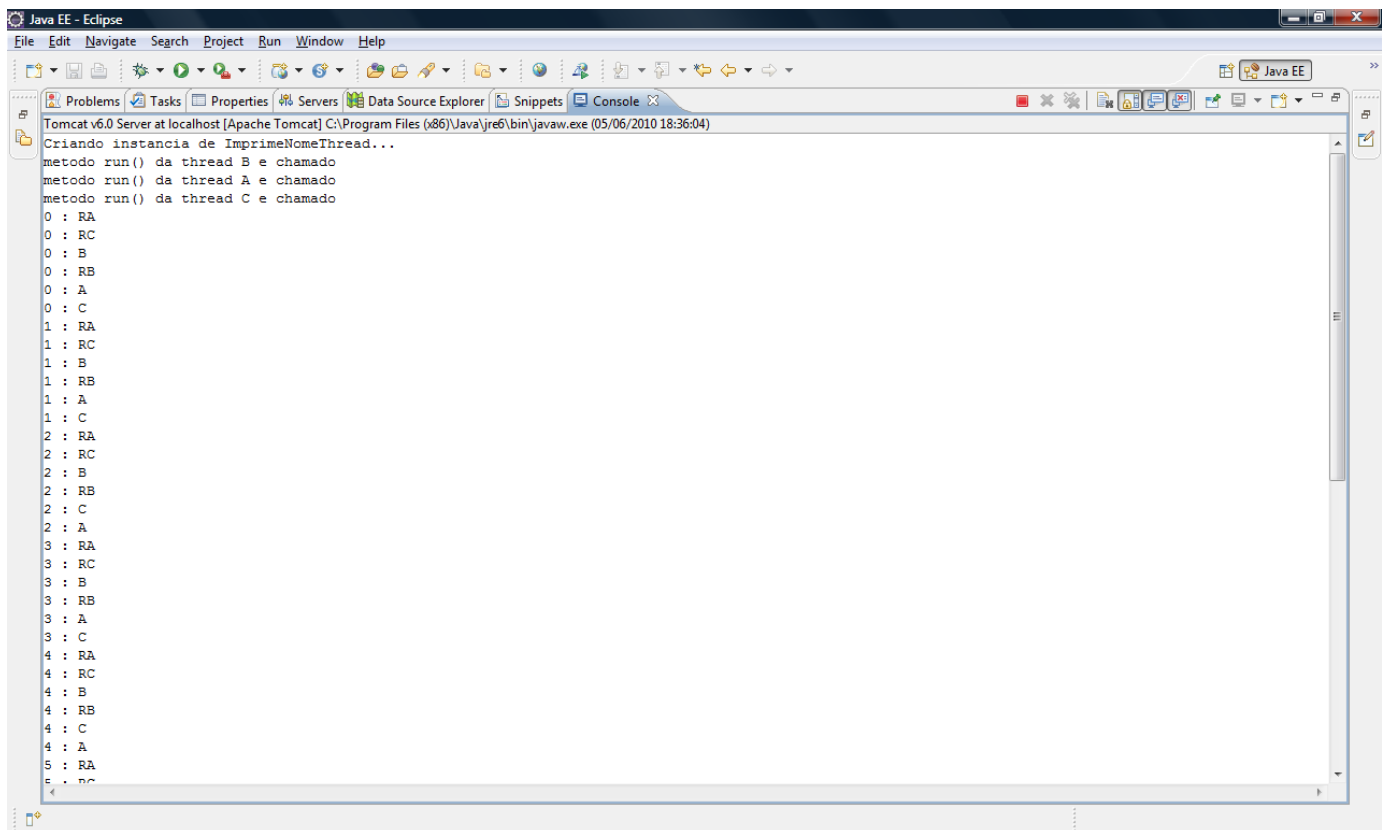
```

2. Altere o código que foi adicionado ao método main no **exercício 2** para que fique conforme abaixo, ou seja, sem ter que declarar um objeto do tipo **Thread** e depois invocar o método **start()** e execute a aplicação para ver o resultado.

```

ImprimeNomeRunnable pntr1 = new ImprimeNomeRunnable("RA");
ImprimeNomeRunnable pntr2 = new ImprimeNomeRunnable("RB");
ImprimeNomeRunnable pntr3 = new ImprimeNomeRunnable("RC");

```



Exercício 3 - Sincronizando threads

3.1 Threads não sincronizados

1. Crie a classe **DuasStrings.java** que será utilizada para imprimir valores e para a thread em execução.

```

public class DuasStrings {

    // esse método não está sincronizado

```

```

static void print(String str1, String str2) {

    System.out.print(str1);
    try {
        // interrompe a thread
        Thread.sleep(500);
    } catch (InterruptedException ie) {
    }
    System.out.println(str2);
}
}

```

2. Escreva a nova classe de thread implementando a interface Runnable conforme abaixo.

```

public class ImprimeStringsThread implements Runnable {

    Thread thread;

    String str1, str2;

    ImprimeStringsThread( String str1, String str2 ) {

        this.str1 = str1;
        this.str2 = str2;
        thread = new Thread(this);
        thread.start();
    }

    public void run() {

        DuasStrings.print(str1, str2);
    }
}

```

3. Crie a classe abaixo e execute como uma aplicação Java e veja o resultado na console do Eclipse. Este exemplo mostra resultados não desejados uma vez que o thread não foi sincronizado.

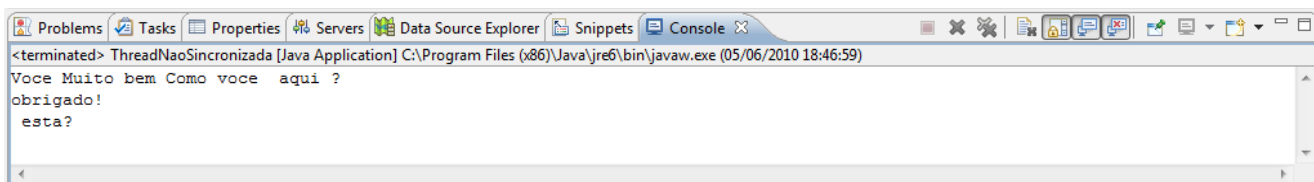
```

public class ThreadSincronizada {

    public static void main(String[] args) {

        new ImprimeStringsThread("Voce ", " aqui ?");
        new ImprimeStringsThread("Muito bem ", "obrigado!");
        new ImprimeStringsThread("Como voce ", " esta?");
    }
}

```



3.2 Sincronizando threads através de métodos

1. Para sincronizar threads é necessário utilizar **synchronized**, então altere a classe **DuasString.java** conforme código abaixo fazendo com que o método **print** seja sincronizado.

```

public class DuasStrings {

```

```
// este método não está synchronized
synchronized static void print(String str1, String str2) {

    System.out.print(str1);
    try {
        // interrompe a thread
        Thread.sleep(500);
    } catch (InterruptedException ie) {
    }

    System.out.println(str2);
}
}
```

3.3 Sincronizando threads através de “sentenças”

1. Modifique a classe **ImprimeStringsThread.java** adicionando mais um atributo e um novo construtor, fazendo sobrecarga de construtores na classe, conforme na listagem abaixo:

```
DuasStrings ds = new DuasStrings();

ImprimeStringsThread(String str1, String str2, DuasStrings ds) {
    this.str1 = str1;
    this.str2 = str2;
    this.ds = ds;
    thread = new Thread(this);
    thread.start();
}
```

2. Agora modifique o método **run()** para que fique conforme o código abaixo:

```
public void run() {
    //sincronizando sobre objeto DuasStrings
    synchronized(ds) {
        ds.print(str1, str2);
    }
}
```

3. Execute novamente a classe **ThreadSincronizada.java** e veja o resultado na console do **Eclipse**.